Analysis and Design of Algorithms

IMCG F-6/2
Islamabad

Fall 2024 Semester
BS course: CS 311 Analysis and Design of Algorithms

Course Instructor**: Saqib Iqbal**

Defining Computer Science

- Computer science is the study of the storage, transformation and transfer of information. The field encompasses both the *theoretical study* of algorithms (including their design, efficiency and application) and the *practical problems* involved in implementing them in terms of computer software and hardware

  (The Linux Information Project)

History

- Procedures for solving geometric and arithmetic problems were formulated by ancient Greeks
- Some two thousands years ago, the celebrated procedure for finding greatest common divisor (gcd) was discovered by *Euclid*
- The word *algorithm* comes from the name of the 9th century Persian mathematician *Abu Abdullah Muhammad ibn Musa al-Khwarzimi*
- His works introduced algebraic concepts
- He worked in Baghdad at the time when it was the centre of scientific studies and trade.
- Al-Khwarzimi's name was translated into Latin, and eventually became *algorithm*

Algorithms today

- The word originally referred only to the rules of performing arithmetic
- The word evolved to include all definite procedures for solving problems or performing tasks
- In the mid-twentieth century *D.E. Knuth* undertook in depth study and analysis of algorithms
- His work is embodied in his comprehensive book '*The art of computer programming*', which serves as a foundation for modern study of algorithms

Definition

- An algorithm is an orderly step-by-step procedure, which has the characteristics:

  1) It accepts one or more input value
  2) It returns at least one output value
  3) It terminates after finite steps

- An algorithm may also be viewed as a tool for solving a computational problem

'Determine whether the number *x* is in the list *S* of *n* numbers. The answer is *Yes* if *x* is in *S* and *No* if it is not'

S = [5, 7, 11, 4, 9] n = 5 x = 9 is an instance of the problem

Solution to this instance is 'Yes'

Applications

- Algorithms have been developed to solve an enormous variety of problems in many application domains. Some broad categories of algorithms are listed below.

  - Sorting Algorithms
  - Searching Algorithms
  - String Processing (Pattern matching, Compression, Cryptography)
  - Image Processing (Compression, Matching, Conversion)
  - Mathematical Algorithms (Random number generator, matrix operations)

Applications (contd…)

- Some applications do not explicitly require algorithmic content at the application level
- Even so, it may rely heavily upon algorithms
- Does the application require fast hardware?
  - Hardware design uses algorithms
- Does the application rely on networking?
  - Routing in networks relies heavily on algorithms
- Does it use a language other than machine code?
  - Compilers, Interpreters make extensive use of algorithms

Analysis of Algorithms

- Analyzing an algorithm has come to mean predicting its performance and the resources that it requires
- The purpose of algorithm analysis is to determine:
  - Time efficiency
    - Performance in terms of running times for different input sizes
  - Space utilization
    - Requirement of storage to run the algorithm
  - Correctness of algorithm
    - Results are trustworthy, and algorithm is robust

Algorithm Efficiency

- Time efficiency remains an important consideration when developing algorithms
- Often performance draws line between feasible and infeasible
- Algorithms designed to solve the same problem may differ dramatically in efficiency
- These differences can be much more significant than differences due to hardware and software
- Example : Sequential search vs. Binary search

Algorithm Efficiency (contd…)

- The number of comparisons done by sequential search and binary search when *x* (value being searched) is larger than all array items

Array Size	Number of comparisons - Sequential search	Number of comparisons - Binary search
128	128	8
1,024	1,024	11
1,048,576	1,048,576	21
4,294,967,296	4,294,967,296	33

Algorithm Efficiency (contd…)

- Another comparison in terms of algorithm execution time

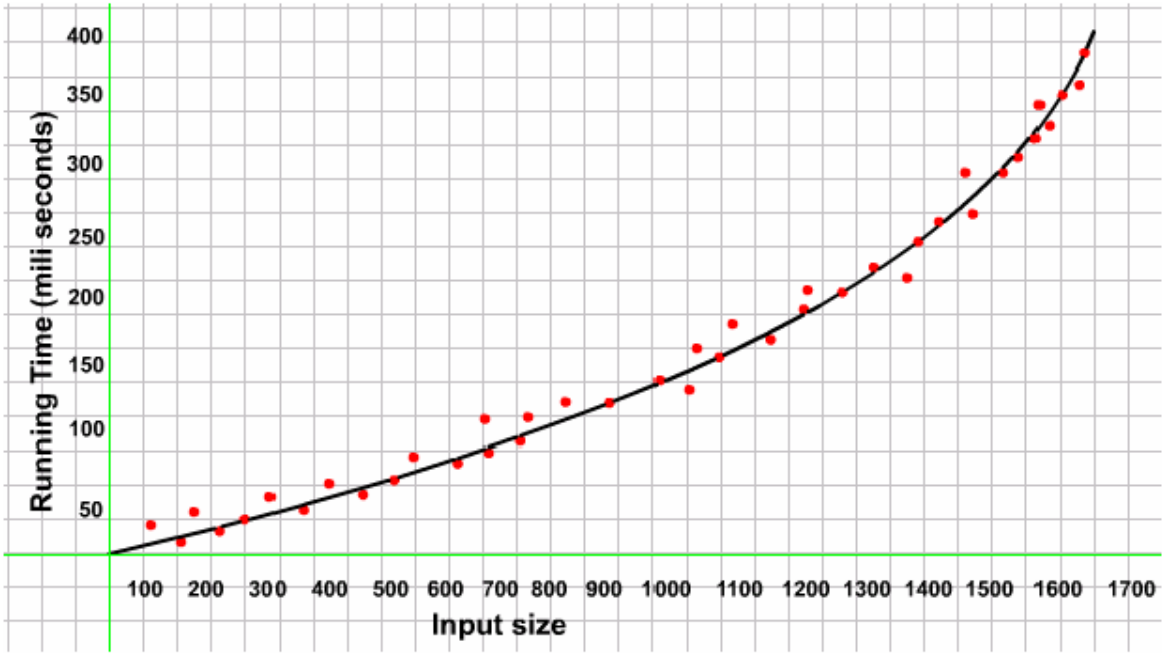
Execution time of Algorithm 1	Execution time of Algorithm 2
41 ns	1048 µs
61 ns	1s
81 ns	18 min
101 ns	13 days
121 ns	36 years
161 ns	$3.8 * 10^7$ years
201 ns	$4 * 10^{13}$ years

Approaches to Analysis

- Basically three approaches can be adopted to analyze algorithm *running time* in terms of *input size:*
  - Empirical Approach
    - Running time measured experimentally
  - Analytical Approach
    - Running time estimated using mathematical modeling
  - Visualization
    - Performance is studied through animation for different data set**s**

Empirical Approach

- The running time of algorithm is measured for different data sizes and time estimates are plotted against the input. The graph shows the trend



Limitations

- Running time critically depends on:
  - Hardware resources used
    - (CPU speed, IO throughput, RAM size)
  - Software environment
    - (Compiler, Programming Language)
  - Program design approach
    - (Structured, Object Oriented)

Analytical Approach

- We want a measure that is independent of the computer, programming language, and complex details of the algorithm
- Usually this measure is in terms of how many times a *basic operation* is carried out for each value of the *input size*
- Strategy: Running time is estimated by analyzing the *primitive operations* which make *significant contributions* to the overall algorithm time. These may broadly include:
  - Comparing data items
  - Computing a value
  - Moving a data item
  - Calling a procedure

Algorithm Specification

- Plain natural language
  - High level description
- Pseudo Code
  - Low level to facilitate analysis and implementation

Specification Using Natural Language

- Preorder Tree Traversal Algorithm

  Step1. Push tree root to stack

  Step2. Pop the stack. If stack is empty exit, else process the node

  Step3. Travel down the tree following the left most path, and pushing each right child onto the stack

  Step4. When leaf node is reached, go back to step 2.

Pseudo Code Convention

Declaration: Algorithm procedure name with parameters e.g MERGE(A,p,q)
Assignments : Using left arrows (←) e.g j← k ← p
Comparisons : Using symbols ≤ ≥ ≠ = > <
Logical : Using connectives **and** , **or**
Computations: Using arithmetic symbols
Exchanges: Using symbol ↔ e.g A[i] ↔ A[k] (swap)
Loops:

for ---- do ----, **for-----downto ---do---**
while ---- do---
do ----- until---

Conditions: **if ---- then-----else-----**
Block structure: Using indentation

```
do---
    do----
        if--- else ----
```

Comments: Using symbol ▶

Example

```
1   for j ← 2 to n
2     do key ← A[j]
3       ▶Insert A[j] into sorted sequence A[1..j-1]
4         i ← j ← 1
5         while i > 0 and A[i] > key
6               do A[i+i] ← A[i]
7                  i ← i-1
8         A[i+1] ← key
```

Algorithm Design

- There are many approaches to designing algorithms:
  - Divide-and-Conquer
  - Greedy
  - Dynamic Programming
  - Brute Force
  - Approximation
- Each has certain advantages and limitations

Course Outline

- Aims
  - To enable students to analyze the complexity of algorithms, and thus equip them with the skills to compare various algorithms for a problem and evaluate which one to use under given conditions.
  - To familiarize them with algorithms for well known problems through a detailed discussion on these algorithms.
  - To enable students to appreciate the role of algorithms in different application areas e.g. data mining, high performance computing.
  - To introduce students to current research in selected application areas.

Course outline (contd…)

Session	Lecture	Readings
1-3	Introduction: Introduction to algorithmic analysis, Mathematical preliminaries, asymptotic notation and analysis	Chapter 1-3
4-8	Review of Sorting Algorithms: Insertion Sort, Quick Sort, Merge Sort, Heap sort, Sorting in linear time	Chapter 6-8
9-11	Review of Searching and Tree Structure Algorithms: Linear and Binary search, Hashing, Red-Black trees	Chapter 11,12
12	Sessional	
13-16	Graph Algorithms: Graph representation, Bread-First and Depth-First search, Minimum Spanning Tree, Shortest Path	Chapter 22-25

Course outline (contd…)

17-19	Dynamic Programming: Assembly line scheduling, Matrix chain multiplication, Longest common subsequence	Chapter 15
20	String Matching Algorithms: Naive string matching algorithm, String matching with finite automata	Chapter 32
21	Greedy Algorithms: Greedy Approach, 0/1 Knapsack problem, Huffman codes, Activity selection	Chapter 16
22	Sessional 2	
23-26	Advanced topics: Introduction to NP-completeness, proofs and problems, approximation algorithms	Chapter 34, 35
27-32	Algorithms in various application areas	

Course outline (contd…)

- **Evaluation Criteria:**
  - Final exam 50%
  - Sessionals 25%
  - Assignments+Quizzes 25%

- **Recommended Readings:**
  - T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, McGraw Hill, Second Edition, 2001
  - R. Sedgewick and P. Flajolet, *An Introduction to the Analysis of Algorithms*, Addison Wesley, 1996
  - M.A. Weiss, *Data Structures & Algorithm Analysis in C++*, Addison Wesley, 2nd Edition, 2004
  - D.E. Knuth, *The Art of Computer Programming*, Addison Wesley, 1997

Course outline (contd…)

- Policies
  - Late assignments may be accepted with marks reduction. There will be a 10% reduction for assignments submitted up to 24 hours late
  - Students who have copied assignments or whose assignments have been copied will both be given a zero
  - Plagiarism is not acceptable. Anyone found to be guilty of plagiarism in an assignment will be given a zero in that assignment
  - Quizzes may be unannounced or announced (depends on your response!)
- Website